

## 7. APPROXIMATION TECHNIQUES IN SYMBOLIC CIRCUIT ANALYSIS

Francisco V. Fernández

*Dept. Electronics and Electromagnetism, University of Sevilla and IMSE-CNM, CSIC*

Carlos Sánchez-López

*IMSE-CNM, CSIC and University of Sevilla, Spain, and University of Tlaxcala, México*

Rafael Castro-López

*IMSE-CNM, CSIC and University of Sevilla*

Elisenda Roca-Moreno

*IMSE-CNM, CSIC and University of Sevilla*

**Abstract:** Symbolic circuit analysis suffers from the exponential growth of expression complexity with circuit size. Therefore, either if the symbolic expressions are used for gaining insight into circuit operation or for repetitive computer-based evaluations, simplification becomes mandatory. This chapter reviews the different existing techniques for symbolic expression simplification, classifying them into three categories according to the step at which the simplification is performed: on the circuit equations, during the solution of the circuit equations or after the circuit equations have been solved. Pros and cons of each approach are discussed.

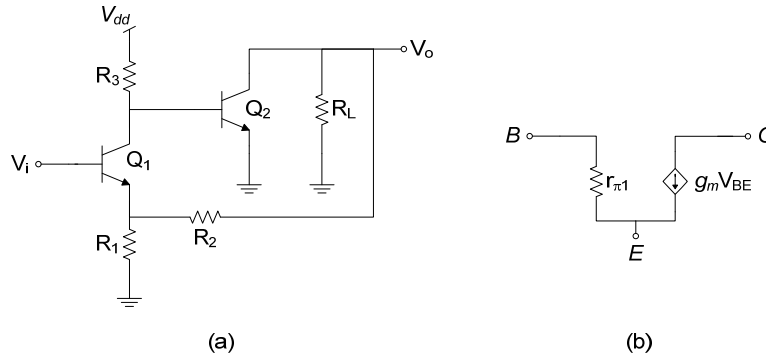
### 1. Introduction

Symbolic analysis refers to circuit analysis in which all or part of the circuit parameters are kept as symbols. Symbolic analysis tools have been typically restricted to linear models, therefore limiting their application to linear(ized) or weakly non-linear analysis.

In this chapter we will concentrate on small-signal frequency domain analysis. We can distinguish between fully symbolic analysis and semi-symbolic analysis. In the former case, all component parameters and the complex frequency variable are maintained as symbols. In the latter, some or all component parameters are given numerical values and the rest are maintained as symbols. Our discussion will focus on fully symbolic analysis, since, from the perspective of this chapter, semi-symbolic analysis is considered a particular case of the former.

The analysis of an amplifier like that in Fig. 1 provides a symbolic expression of the voltage gain with 21 terms:

$$H(s, \mathbf{x}) = \frac{g_{m1}g_{m2}r_{\pi1}r_{\pi2}R_3R_L(R_1+R_2)+R_1R_L(R_3+r_{\pi2})g_{m1}r_{\pi1}+R_1R_L(R_3+r_{\pi2})}{g_{m1}g_{m2}r_{\pi1}r_{\pi2}R_3R_LR_1+g_{m1}r_{\pi1}(R_2+R_L)(R_3+r_{\pi2})R_1+(R_2+R_L)(R_3+r_{\pi2})(R_1+r_{\pi1})+(R_3+r_{\pi2})r_{\pi1}R_1} \quad (1)$$



**Figure 1:** (a) Feedback amplifier; (b) BJT small-signal model.

Although this expression is relatively simple, the operation as a feedback amplifier can be better understood if we consider that the first term in numerator and denominator is much larger than the other ones, and if they are neglected, the approximated voltage gain becomes an expression so much easier to interpret:

$$H(s, \mathbf{x}) = \frac{R_1 + R_2}{R_1} \quad (2)$$

Moreover, in case that this expression is to be used in some kind of computer application that requires repetitive evaluations, this can be done much more efficiently.

However, fully symbolic circuit analysis suffers from the exponential growth of expression complexity with circuit size. As Section 4 will show, the number of terms in the symbolic expression grows to about  $10^4$  terms for a Miller two-stage amplifier with 7 transistors and to more than  $10^{10}$  terms for an opamp with about 20 transistors. This expression complexity not only hampers its interpretation or repetitive evaluation but it can even make their generation computationally impossible. Therefore, the introduction of simplification techniques in symbolic circuit analysis becomes mandatory.

Considering the step of the analysis process at which the simplification is performed we will distinguish three types of techniques:

1. *Simplification before generation (SBG) techniques.* The simplification is performed at the circuit model, graph or matrix level, directly on the graph or matrix representing the circuit equations. The goal is to obtain a simplified graph or matrix that can be solved much more efficiently and that yields much simpler symbolic results with a controlled error.
2. *Simplification during generation (SDG) techniques.* The simplification is applied during the solution process of the circuit equations. The goal is to produce only the most significant part of the symbolic expression, without

wasting time in generating symbolic terms with little influence on the final results.

3. *Simplification after generation (SAG) techniques.* The simplification is performed directly on the symbolic solution. Therefore, it requires the previous circuit analysis and solution of the circuit equations.

Following this classification of techniques, the three remaining sections in this chapter are devoted to them.

## 2. Simplification after generation techniques

As stated above, simplification after generation techniques are applied on the symbolic network functions, once the circuit has been symbolically analyzed. In case these techniques are combined with other simplification (SBG and/or SDG) techniques, they would be the last ones to be applied, but they were the first ones that chronologically appeared, and for this reason, they will be discussed first.

SAG techniques are typically applied on network functions in expanded format:

$$H(s, \mathbf{x}) = \frac{\sum_{j=1}^m s^j f_j(\mathbf{x})}{\sum_{i=1}^n s^i g_i(\mathbf{x})} \quad (3)$$

where the coefficients of the different powers of the complex frequency are sums-of-products of symbolic circuit parameters:

$$\mathbf{x}^T = \{x_1, x_2, \dots, x_Q\} \quad (4)$$

Reported simplification criteria consider simplification at the full frequency range, i.e., the simplification is performed for each coefficient of the complex frequency in numerator and denominator of (3). Let us denote

$$h_k(\mathbf{x}) = \sum_{l=1}^T h_{kl}(\mathbf{x}) \quad (5)$$

as any coefficient  $f_i(\mathbf{x})$  or  $g_j(\mathbf{x})$  in (3). The simplification is usually performed by heuristically pruning the insignificant terms in each coefficient  $h_k(\mathbf{x})$  of the complex frequency variable in (3) so that an approximate polynomial,  $h_{kA}(\mathbf{x})$ , is found for each coefficient. This approximate polynomial fits the original one within a user-specified maximum error parameter  $\varepsilon_M$  inside a given region  $R$  of the symbolic parameter space:

$$\max_{\mathbf{x} \in R} \left| \frac{h_k(\mathbf{x}) - h_{kA}(\mathbf{x})}{h_k(\mathbf{x})} \right| < \varepsilon_M \quad (6)$$

Most of the reported approaches perform this fitting only at a single point of the parameter space  $\mathbf{x}_o$ , commonly called the *nominal* or *design* point.

The simplification criterion in [1] looks for the largest magnitude term for each coefficient  $h_k(\mathbf{x}_o)$  and multiplies it by a user-defined maximum error  $\varepsilon_o$ , that defines a discrimination threshold. Then, all the terms are taken one by one and those whose magnitude is below the calculated threshold are eliminated. In other words, any term is eliminated from (5) if it fulfills the following condition:

$$|h_{kl}(\mathbf{x}_o)| < \varepsilon_o \cdot \max(|h_{k1}(\mathbf{x}_o)|, |h_{k2}(\mathbf{x}_o)|, \dots, |h_{kT}(\mathbf{x}_o)|) \quad (7)$$

Its main drawback is the lack of control on the accumulated error for each coefficient—the accumulated value of the deleted terms can represent either a small or large part of the total magnitude of each coefficient. Consequently, coefficient errors will probably differ considerably for different coefficients, and large magnitude and phase errors and large pole/zero displacements can be thus expected.

More elaborated criteria require the previous sorting of terms in  $h_k(\mathbf{x})$  according to their magnitude at the nominal point  $\mathbf{x}_o$ . One possibility is to eliminate the  $P$  smallest magnitude terms,  $P$  being the largest integer for which the accumulated error is below  $\varepsilon_M$  [2]–[4]:

$$\frac{\left| \sum_{l=1}^P h_{kl}(\mathbf{x}_o) \right|}{\left| \sum_{l=1}^T h_{kl}(\mathbf{x}_o) \right|} < \varepsilon_M \quad (8)$$

Mutually canceling terms do not contribute to (8) because they are added with their respective signs. However, such terms may become significant when the simplified formula is evaluated at points other than  $\mathbf{x}_o$ . Hence, although this criterion gives very accurate results at  $\mathbf{x}_o$ , the resulting error at other points may be well beyond  $\varepsilon_M$ . This happens, for instance, when mismatches among nominally matched devices are taken into account. Such mismatches have a strong influence on characteristics that rely largely on cancellations, such as the common-mode rejection ratio and the power-supply rejection ratio. In these cases, large insight is gained if explicit mismatch parameters are introduced.

One solution to avoid elimination of mutually canceling terms is to modify (8) as follows:

$$\frac{\sum_{l=1}^P |h_{kl}(\mathbf{x}_o)|}{\sum_{l=1}^T |h_{kl}(\mathbf{x}_o)|} < \varepsilon_M \quad (9)$$

Neglected terms that are of the same order of magnitude as the last one remaining in the simplified expression are recovered and kept in this expression [5].

In the previous criteria, if the same error  $\varepsilon_M$  were exactly obtained for all numerator and denominator coefficients in (3), the simplified expression would become

$$H(s, \mathbf{x}) = \frac{(1 - \varepsilon_M) f_0(\mathbf{x}) + s(1 - \varepsilon_M) f_1(\mathbf{x}) + \dots + s^m (1 - \varepsilon_M) f_m(\mathbf{x})}{(1 - \varepsilon_M) g_0(\mathbf{x}) + s(1 - \varepsilon_M) g_1(\mathbf{x}) + \dots + s^n (1 - \varepsilon_M) g_n(\mathbf{x})} \quad (10)$$

where there is no change in magnitude or phase and neither zero nor pole displacement. However, expression simplification is a discrete process and, hence, the actual errors are different for each coefficient:

$$H(s, \mathbf{x}) = \frac{(1 - \varepsilon_{M0n}) f_0(\mathbf{x}) + s(1 - \varepsilon_{M1n}) f_1(\mathbf{x}) + \dots + s^m (1 - \varepsilon_{Mmn}) f_m(\mathbf{x})}{(1 - \varepsilon_{M0d}) g_0(\mathbf{x}) + s(1 - \varepsilon_{M1d}) g_1(\mathbf{x}) + \dots + s^n (1 - \varepsilon_{Mnd}) g_n(\mathbf{x})} \quad (11)$$

This may lead to significant root displacements in circuits where the roots are very sensitive to coefficient variations.

Different solutions have been proposed to overcome these problems. A trivial strategy adds a numerical fitting factor to each simplified coefficient such that the numerical evaluation of the pruned coefficients at the nominal point is made equal to the original ones [4]. This approach has been implemented in different symbolic analyzers and guarantees accuracy at the nominal point, but it does not imply any improvement for points other than nominal.

Another possibility is to monitor the magnitude of each term within each  $h_k(\mathbf{x})$  in (3), to avoid eliminating those whose magnitude is greater than the denominator in (9) [2]. Another approach is to use an adaptive  $\varepsilon_M$  scheme: term-pruning is performed step by step and the pole/zero displacements are monitored at each step so that simplifications can be stopped when such displacements are beyond a user-specified safety margin [3]. Unfortunately, even though this guarantees a low error at the nominal point, it does not ensure good results for different points of the parameter space.

All these criteria have assumed that the frequency remains a symbol. If the system function has to be approximated for a single value of the frequency,  $f_o$ , then it must be evaluated for  $s = j2\pi f_o$ . The problem then reduces to approximating the real and imaginary parts of numerator and denominator and, hence, conceptually it is no different from the approximation of individual coefficients of the network function.

Expression approximation for a bound frequency range using a nominal value approach is not easy. One possibility is to perform the approximation for different frequency points within the given range and include in the final expression every term that is present at least in the simplified expression at one frequency point. Full accuracy, however, is not guaranteed with this approach. Increasing the number of sample frequency points diminishes the likelihood of errors but also diminishes the speed of the algorithm. Sections 3 and 4 will discuss a possible solution to this problem. Although the techniques discussed there can also be directly applied, they will not be discussed here as they have been reported for SBG and SDG approaches.

All previous algorithms perform the approximation at a nominal point and there is no guarantee that the accuracy is high enough at other design points. Parameter variations in practical circuits are usually restricted to bounded regions of the parameter space. One possibility to extend the validity range of the simplified expressions is the repetitive application of any previous criteria to each point (a sufficiently fine grid should be defined) inside the bounded region. However, because dimensions of the parameter spaces of practical circuits are usually very large, this approach is computationally intractable in practice.

A solution was reported in [6]. It is based on the use of ranges of variation [7], i.e. it assumes that each symbol (device model variable, product of variables, or sum of products) may take any value inside a given range of variation:

$$y_i \in [y_{iL}, y_{iH}] \quad (12)$$

where  $y_{iL}$  and  $y_{iH}$  are real numbers and  $y_{iL} \leq y_{iH}$ .

In this approach, the  $P$  least significant terms of each coefficient  $h_k(\mathbf{x})$  are eliminated as long as the following condition is satisfied:

$$\frac{U([A_{cL}, A_{cH}])}{L([S_L, S_H])} < \varepsilon_M \quad (13)$$

where  $[S_L, S_H]$  represents the range of the sum of all terms in coefficient  $h_k(\mathbf{x})$ ,  $[A_{cL}, A_{cH}]$  represents the range of the sum of all terms to be pruned and,  $U(\cdot)$  and  $L(\cdot)$  are the upper and lower range operators, that return the upper and lower limit of the included range, respectively.

To apply this definition, operators among ranges have to be defined:

- **Product of ranges.** Given the multiplication of two symbolic factors,  $y_i$  and  $y_j$ , the range of their product is:

$$y_i y_j \in [\min(y_{iL} y_{jL}, y_{iL} y_{jH}, y_{iH} y_{jL}, y_{iH} y_{jH}), \max(y_{iL} y_{jL}, y_{iL} y_{jH}, y_{iH} y_{jL}, y_{iH} y_{jH})] \quad (14)$$

- **Addition of ranges.** For a given sum of two symbols,  $y_i$  and  $y_j$ , the range of the sum is computed by adding the corresponding bounds of the addends:

$$y_i + y_j \in [y_{iL} + y_{jL}, y_{iH} + y_{jH}] \quad (15)$$

- **Modulus of ranges.** For a given symbolic parameter, product of symbols, or sum of products for which a range  $[y_{iL}, y_{iH}]$  is defined or calculated, the modulus of ranges operator yields another range, defined from the previous one by taking the modulus of the extremes in an appropriate order:

$$[|y_{iL}|, |y_{iH}|] = [\min(|y_{iL}|, |y_{iH}|), \max(|y_{iL}|, |y_{iH}|)] \quad (16)$$

- **Reciprocal of a range.** For a given symbol  $y_i$  whose range does not include zero, its reciprocal is defined as

$$\frac{1}{y_i} \in \left[ \frac{1}{y_{iH}}, \frac{1}{y_{iL}} \right] \quad (17)$$

The major drawback of the variation range technique is that excessively conservative results can be obtained, due to two reasons:

- Some circuit parameters are correlated with each other. If each one is assigned a range, range operators ignore the correlations, and this yields overestimation of the real range.
- Direct substitution of the symbolic parameters with their ranges and the real arithmetic operators with their corresponding interval arithmetic operators yield the so-called natural interval extension of the symbolic polynomials. Its main drawback is that the width of the range may be considerably larger than the real one, because each range bounds may be calculated with the upper range bound of one symbolic parameter at a polynomial term and the lower bound at another. This problem is partially avoided by using other interval extensions, like the mean value interval extension of a function [7]  $f(\mathbf{x})$ :

$$F_{MV}(\mathbf{X}) = f(\mathbf{m}) + \sum_{i=1}^n D_i F(\mathbf{X})(X_i - m_i) \quad (18)$$

where capital letters denote interval extension, the set  $\mathbf{m} = \{m_1, m_2, \dots, m_n\}$  is the vector of mean values of the variables  $x_i$ , and  $D_i F(\cdot)$  is the interval extension of the function derivative with respect to the  $i$ -th variable. This

interval extension can be computed using the natural interval extension of the derivatives, or recursively calculated using (18).

The approximation based on ranges of variation becomes especially interesting in case of matching devices. Matching devices are those elements designed to have identical nominal values, e.g. differential pairs and current mirrors. Due to process variations mismatches occur between such elements. In some performance characteristics device mismatch plays a dominant role and, therefore, introducing explicit mismatch parameters becomes extremely convenient, e.g., given two matched transistors,  $M_1$  and  $M_2$ , their transconductance is symbolically represented as:

$$g_{m1} \rightarrow g_m + \Delta g_m \quad g_{m2} \rightarrow g_m - \Delta g_m \quad (19)$$

In nominal value approaches [2],[3] mismatch parameters, e.g.  $\Delta g_m$ , are handled like any other symbolic parameters, i.e., a numerical value equal to the maximum mismatch value is assigned to such mismatch parameter for approximation purposes. However, this does not correspond to the basic philosophy of a mismatch parameter, as there is no mismatch value that can be assimilated to a nominal value, and even the sign is unknown. But the concept of a mismatch parameter, i.e., a parameter that can take any value between a minimum and maximum values fits perfectly with the range of variation philosophy.

### 3. Simplification before generation techniques

In this kind of techniques, approximations are performed directly on the network equations, either in the form of a matrix, a graph, or the small-signal equivalent circuit itself, which are all mainly determined by the solution technique for the network equations that will be applied afterwards.

The complexity reduction of the network equations has a tremendous impact on the complexity of the symbolic results. Therefore, the degree of interpretability introduced is significantly improved. Besides, the computation time and memory requirements of the subsequent SDG/SAG algorithms are considerably reduced due to the exponential relationship between circuit complexity and the number of terms to be generated.

The basic goals of a simplification before generation approach are:

- The simplified system of equations must model the circuit behavior correctly within the error constraints in the specified frequency range.
- The system of equations resulting after the simplification step should be the simplest possible.

To this end two elements are needed:



- An ordering mechanism for the contribution of the different graph branches/nodes or matrix entries appearing in the graph/matrix that represents the system of equations.
- A stopping criterion to decide the number of devices/nodes or matrix entries that can be affected by the simplification without exceeding the error specifications within the defined frequency range.

According to the form of circuit model or equations in which SBG is applied, the next three subsections will deal with matrix-based, graph-based and circuit-based approaches respectively. In Section 3.4 pros and cons of these approaches are discussed and their error control mechanisms are introduced in Section 3.5.

### 3.1. Matrix-based approaches

A simplification before generation technique at the matrix level was introduced in [8]-[10]. This technique builds the circuit equations through the indefinite nodal admittance matrix:

$$\mathbf{I} = \mathbf{Y} \cdot \mathbf{V} \quad (20)$$

The network function can be expressed as the ratio of two cofactors (a first-order one and a second-order one) of this matrix. As a first step, device parameters are eliminated from each cofactor of the nodal admittance matrix if the error induced in the cofactor is below a given error threshold. The error induced in one matrix determinant by a modification  $\Delta m_{i,j}$  of the value at the location  $(i, j)$  is

$$\Delta(|\mathbf{M}|) = \Delta m_{i,j} \cdot (-1)^{i+j} |\mathbf{M}_{ij}| \quad (21)$$

Matrix  $\mathbf{M}$  in (21) is the appropriate cofactor of the nodal admittance matrix, according to the numerator or denominator of the network function, and  $\mathbf{M}_{ij}$  is the minor, obtained by deleting the  $i$ -th row and the  $j$ -th column of  $\mathbf{M}$ .

As each circuit element is mapped into four positions on the nodal admittance matrix, parameter elimination can be performed at one, two, or all four positions.

Concurrently with the device parameter elimination, this technique tries to reduce determinant dimension by factoring out rows and columns with only one nonzero entry and performs row and column operations to reduce the number of symbols or nonzero entries. These heuristics do not alter the value of the determinant and are not approximations in fact. However, they are valuable as they partially palliate the cancellation problem of determinant-based approaches.

Capacitors must also be included in the nodal admittance matrix for high-frequency analysis. Then, applied approximations are only valid in a limited frequency range. Three scanning frequencies are considered per decade, used to evaluate the influence of elimination of the corresponding capacitor parameter. One element is eliminated at

four, two, or one location, only if the induced error in the determinant is smaller than an error bound for all scanning frequencies.

This approach separately controls the error in the determinants that correspond to the numerator and the denominator of the network function but not in the network function itself. If we consider a generic matrix formulation:

$$\mathbf{A} \cdot \mathbf{x} = \mathbf{b} \quad (22)$$

a simple solution would be to calculate the effect on the elements of interest in vector  $\mathbf{x}$  when any matrix entry is removed. This involves a numerical matrix inversion for each removal that is tried.

A nice solution to avoid such repetitive calculation of inverse matrices is to apply the Sherman-Morrison formula that allows calculating the numerical influence in the system output when a matrix entry  $a_{i,j}$  is modified by a value  $\Delta a_{i,j}$  [11]:

$$a_{i,j}^* = a_{i,j} + \Delta a_{i,j} \quad (23)$$

The perturbation on the  $k$ -th element of vector  $\mathbf{x}$  can be calculated as [11]:

$$\Delta x_k = \frac{-\Delta a_{ij} \cdot a_{ki}^{(-1)}}{1 + \Delta a_{ij} \cdot a_{ji}^{(-1)}} \mathbf{A}_{j,\cdot}^{-1} \mathbf{b} \quad (24)$$

where  $a_{ji}^{(-1)}$  is the matrix entry at position  $(j,i)$  of matrix  $\mathbf{A}^{-1}$  and  $\mathbf{A}_{j,\cdot}^{-1}$  is the  $j$ -th row of  $\mathbf{A}^{-1}$ .

According to these perturbation values, a ranking list is generated where each symbol of the matrix is ordered depending on their numerical influence on a requested component (network variable) [11]. Then, starting with the least significant parameter, eliminations are performed while the error keeps below a given threshold. A heuristic to decide if the influence list has to be recalculated is used to avoid a direct re-calculation after each symbol elimination (which would mean a large waste of computational time). It also controls that the matrix that will result after the term elimination is not singular.

### 3.2. Graph-based approaches

The simplification before generation methodology in [12] is performed using the two-graph approach and has been implemented in two steps. In the first step, the voltage graph and the current graph are built for both the numerator and the denominator of the network function. Each device contribution (represented in the form of an admittance  $y$ ) to the numerator,  $c(y,N)$ , and denominator,  $c(y,D)$ , are calculated, as well as each device complementary contribution,  $\bar{c}(y,N)$  and  $\bar{c}(y,D)$ :

$$\begin{aligned}
c(y, N) &= \frac{yN(y; )}{N} & \bar{c}(y, N) &= \frac{N( ; y)}{N} \\
c(y, D) &= \frac{yD(y; )}{D} & \bar{c}(y, D) &= \frac{D( ; y)}{D}
\end{aligned} \tag{25}$$

where  $N(y; )$  and  $D(y; )$  correspond to the sum of product terms in  $N$  and  $D$  that contain the admittance  $y$ , and  $N( ; y)$  and  $D( ; y)$  to the sum of product terms in  $N$  and  $D$ , that do not contain  $y$ .

This step is performed for a set of frequency samples. Devices with small values of  $c(y, N)$  or  $c(y, D)$  for all frequency samples are weakly contributing devices and are candidates for deletion in the corresponding graphs. Devices with large values of  $c(y, N)$  or  $c(y, D)$  for all frequency samples are strongly contributing devices and are candidates for contraction of its terminal nodes in the corresponding graphs. When any deletion or contraction is performed, all the remaining contributions are recalculated, and if they change significantly with respect to the original value, the deletion/contraction operation of that element is cancelled.

Then, the graphs are further simplified simultaneously. The dual contraction error,  $e_c(y; H)$ , and the dual deletion error,  $e_d(y; H)$ , of  $H(s)$  with respect to the device  $y$  are defined as [12]:

$$e_c(y; H) = \frac{\frac{N(y; )}{D(y; )} - \frac{N}{D}}{\frac{N}{D}} \quad e_d(y; H) = \frac{\frac{N( ; y)}{D( ; y)} - \frac{N}{D}}{\frac{N}{D}} \tag{26}$$

Those elements with low error value can be deleted/contracted from both the numerator and the denominator without affecting the network function value too much. Again, before definitively performing any operation (deletion or contraction) the remaining dual errors are analysed to avoid any operation that causes a large variation on other contributions.

Due to the separate simplification performed on the numerator and the denominator in the first step, it may happen that some devices are eliminated in the numerator (denominator) and still appear at the denominator (numerator). As a result of this, it will be impossible to build a simplified circuit out of the results. Also, the simplification procedure is performed on a set of discrete frequency samples; this means that the accuracy is only guaranteed at those frequency values.

A technique based on a signal flow graph approach was introduced in [13],[14]. This flow graph is composed of voltage nodes (that represent the voltage of the nodes in the circuit) and current nodes (that represent the voltage of the nodes in the circuit multiplied by the sum of all the self-admittances attached to them). Then, the

simplification is performed on this network graph by the application of the following set of operations, called network model transformations [13],[14]:

- *Removal of signal path*: deletion of an incoming meta-edge (set of all parallel branches between two nodes) of a summing vertex.
- *Open-loop root removal*: deletion of a meta-edge not belonging to a signal path.
- *Vertex-pair contraction*: short the voltage vertices and current vertices associated to two nodes.
- *Subgraph substitution*: replacement of a part of the graph by other different graph without altering the connectivity of the network graph.

An important point of the analysis of a linear network,  $N$ , is related to the calculation of the network graph complexity:

$$C(N) = n_{fol} + n_{sum} + n_{fwp} + n_{fbl} \quad (27)$$

where  $n_{fol}$  is the number of open-loop roots,  $n_{sum}$  is the number of summing points,  $n_{fwp}$  is the number of forward paths and  $n_{fbl}$  is the number of feedback loops in the graph.

For each of the four operations described above, taking into account the performance parameters,  $p_i(N, X)$ , a ranking function is defined:

$$R_p(N, T_j, X) = \frac{[E_p(N, T_j, X)]^\alpha}{[Q_p(N, T_j)]^{1-\alpha}} \quad (28)$$

being  $X$  the design point,  $T_j$  the  $j$ -th network model transformation, and

$$Q_p(N, T_j) = C(N) - C(T_j(N)) \quad (29)$$

the transformation quality factor, that measures the difference between the complexity of the original network model to the one after one simplification step  $j$ ,  $\alpha$  is a tuning factor and

$$E_p(N, T_j, X) = |P(N', X) - P(N, X)| \quad (30)$$

is the model performance error, that is the weighted norm of the performance deviation vector. This model performance error has to be measured in a set of frequency points that cover the frequency range defined by the user. According to the ranking value defined in (28), the simplification operations are performed until the

specified errors are fulfilled.

Like the previous algorithm, the simplification is performed on a finite number of frequency samples. No SDG algorithm has been defined for this methodology, and it is not clear that a methodology of this kind is feasible for this type of graphs.

### 3.3. Circuit-based approaches

The approach in [15] performs the approximation on the network under analysis directly at the circuit level. It replaces those elements whose contribution (appropriately measured) to the network function is small, by a zero-admittance (element removal) or zero-impedance element (contraction of terminal nodes).

A first question is to decide if node contractions must be prioritized over branch removals or vice versa. After the SBG process, the resulting simplified graph must be solved, commonly by the application of an SDG process. The most efficient SDG algorithms reported are based on the matroid theory (see Section 4) and their computational complexity grows much faster with the number of circuit nodes than with the number of graph branches. Therefore, node contractions are prioritized in the proposed algorithm. The following steps summarize the algorithm operation:

- a) Compute the contribution to the network function of the contraction of the terminal nodes of each device individually and build a sorted list.
- b) Pick the least significant contraction from the list and compute the error.
- c) If the maximum error has not been exceeded, perform the node contraction, remove all devices connected between that pair of nodes, reorder the contraction list and go to step (b); otherwise continue with the next step.
- d) Compute the contribution to the network function of the removal of each branch individually and build a sorted list.
- e) Pick the least significant branch removal from the list and compute the error.
- f) If the maximum error has not been exceeded, perform the branch removal, reorder the removal list and go to step (d); otherwise end the algorithm.

However, the experience with a large number of circuits shows that the order in which nodes are contracted or branches are removed keeps basically the same as the order in which they appear in the contraction/removal sorted list built at steps (a) and (c). As a consequence of this, the time required to perform the simplification can be highly decreased by eliminating the reordering of such lists in steps (c) and (f), without losing accuracy control.

### 3.4. Comparison of SBG approaches

As stated above, matrix-based approaches are able to perform elimination of matrix entries at one, two or four positions. Elimination at four positions is equivalent to

branch deletion in graph-based and circuit-based approaches. Elimination at one or two positions may provide some extra simplification. However, it decreases the insight on the simplified results since the simplified matrix does not correspond to a circuit with less devices and/or nodes.

The branch contraction operation in graph-based and circuit-based approaches does not have a correspondence on matrix-based approaches. Therefore the complexity of the results after the SBG step may differ between the different techniques from circuit to circuit.

A very important drawback of matrix-based approaches is that the SBG step must usually be followed by a SDG step. The most efficient SDG algorithms reported are based on the two-graph approach (see Section 4), which is obviously not applicable to the results of a matrix-based SBG techniques but it can be applied to the other two categories of techniques.

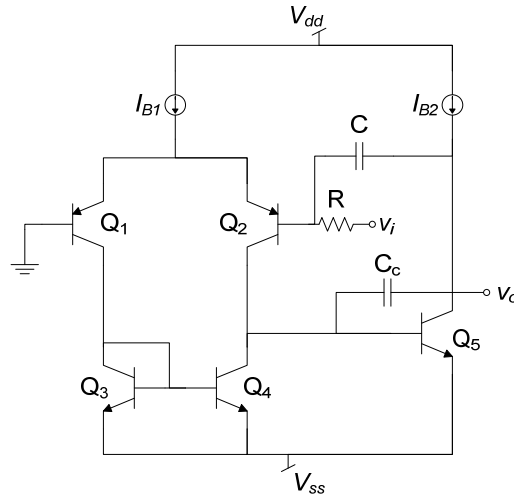
The circuit-based techniques may yield slightly more complex results as no separate deletion/contraction operations in numerator and denominator are performed. However, it exhibits two advantages: first, no graph or matrix for the circuit has to be built and second, during the simplification process, a direct correspondence with the nodes and devices of the original circuit is kept.

### **3.5. Error control**

In all SBG approaches above, an error control mechanism exists that decides when no more matrix entries can be eliminated, no more nodes can be contracted, or no more branches can be deleted.

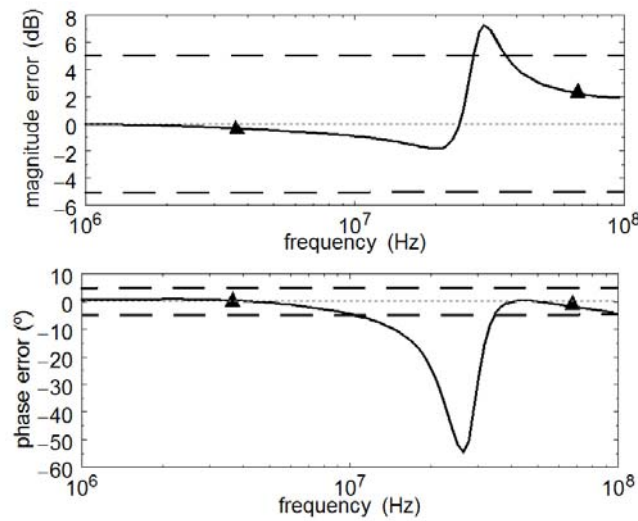
The approaches in [8]-[14] perform the evaluation of the contributions to the network function of the elimination of matrix entries or the successive node contractions and branch removals at a set of frequency samples within the range being considered. However this solution may yield simplified matrices/graphs/circuits that do not fulfil the error specification at some frequency points outside this initial selected set. Besides, a dense sampling, at least in the neighbourhood of the poles and zeros of the system will be necessary because of the significant variations of the network function around these points. Therefore, there are two drawbacks of error control methods based on frequency sampling:

- 1) The correctness (error specs are met in the complete frequency range) of the resulting simplified circuits cannot be guaranteed.
- 2) A dense spectrum of sampled points decreases the possibility of error excess but increases noticeably the computational cost of the algorithm.



**Figure 2:** Miller integrator.

As an illustrative example let us consider the integrator in Fig. 2. Simplification before generation is applied to this circuit with a magnitude and phase error specification of  $\Delta|H| \leq \pm 5\text{dB}$  and  $\Delta\phi_H \leq \pm 5^\circ$  in the frequency range  $1\text{Hz} \leq f \leq 100\text{MHz}$ . The magnitude and phase errors are evaluated at a small set of frequency samples. Fig. 3 shows the resulting magnitude and phase errors. Dashed lines represent the magnitude and phase error specifications. The solid triangles represent some of the frequency samples. As can be seen, error specifications are met at the frequency samples but are considerably exceeded at other frequency values. Increasing the number of frequency samples can palliate the problem, but the appropriate number of frequency samples is not known a priori and the computation time grows linearly with the number of samples.



**Figure 3:** Magnitude and phase errors.

To solve the problems of sampling-based approaches, an interval analysis approach is proposed in [16],[17]. Let us denote  $H_{ex}(s)$  the network function of the complete circuit with only the complex frequency  $s$  as symbolic parameter, and  $H_{ap}(s)$  the analogous network function of a simplified circuit in which the appropriate node contraction(s) and/or device removal(s) have been performed during the application of the SBG algorithm. The magnitude and phase errors are given by:

$$\Delta|H| = \frac{|H_{ex}(j\omega)| - |H_{ap}(j\omega)|}{|H_{ex}(j\omega)|} = 1 - \frac{\sqrt{\frac{N_{apr}^2 + N_{api}^2}{D_{apr}^2 + D_{api}^2}}}{\sqrt{\frac{N_{exr}^2 + N_{exi}^2}{D_{exr}^2 + D_{exi}^2}}} \quad (31)$$

$$\Delta\phi_H = \angle H_{ex}(j\omega) - \angle H_{ap}(j\omega) = \arctan \frac{N_{exi}}{N_{exr}} - \arctan \frac{D_{exi}}{D_{exr}} - \arctan \frac{N_{api}}{N_{apr}} + \arctan \frac{D_{api}}{D_{apr}}$$

where

$$\begin{aligned} H_{ap}(j\omega) &= \frac{N_{ap}(j\omega)}{D_{ap}(j\omega)} = \frac{N_{apr}(j\omega) + jN_{api}(j\omega)}{D_{apr}(j\omega) + jD_{api}(j\omega)} \\ H_{ex}(j\omega) &= \frac{N_{ex}(j\omega)}{D_{ex}(j\omega)} = \frac{N_{exr}(j\omega) + jN_{exi}(j\omega)}{D_{exr}(j\omega) + jD_{exi}(j\omega)} \end{aligned} \quad (32)$$

Therefore, the evaluation of the maximum magnitude and phase errors requires:

- A technique to obtain the network functions  $H_{ex}(s)$  and  $H_{ap}(s)$  of (usually large) analog circuits with only the complex frequency as symbolic variable.
- An efficient technique to obtain the maxima of the functions in (31) when  $\omega$  varies within a given range.

The first requirement can be solved very efficiently by using the numerical interpolation technique [18]. One major problem in polynomial interpolation applied to analog integrated circuits is the dramatic effect of round-off errors, due to the finite precision arithmetics of computers. An efficient adaptive scaling solution to solve this problem is proposed in [19].

A trivial solution to the second requirement is to use a set of frequency samples but the same drawbacks exposed in Section 3.5.1 will also appear here. A possible solution is the application of the Newton-Raphson method to find the zeros of the derivatives of (31). However, since the frequency ranges required for symbolic analysis vary over a wide range the search can be slow, some zeros may be skipped or convergence may fail.

A possible solution to this issue is to resort to the interval analysis techniques that were introduced in Section 2.  $\Delta|H|$  and  $\Delta\phi_H$  in (31) are functions in  $\omega$  that can take



any value within the frequency interval  $[\omega_L, \omega_U]$ . The problem can be solved if accurate estimates of the lower and upper bounds of  $\Delta|H|$  and  $\Delta\phi_H$  when the frequency varies in this range can be calculated. As stated in Section 2, the natural interval extension usually overestimates the range of this function [7]. To avoid this, the natural interval extension is not applied to (31) but to the derivatives of (31). Although, the estimates of the derivatives are also very conservative, the zero inclusion in the resulting interval extension is enough to delimit frequency sub-ranges in which the maximum magnitude and phase errors occur. Then, the exact frequency points for which the maximum magnitude or phase errors occur in those frequency sub-ranges are easily calculated by means of the Newton-Raphson method.

This error evaluation technique has been applied to the SBG techniques in Section 3.3. Initially a very small set of frequency samples is selected. For this set of values, the simplification process described in Section 3.3 is performed. Then, the error evaluation technique in this Section is applied to find any possible violation of error constraints within the full frequency range of interest. If the errors are fulfilled, the procedure stops. Otherwise, the frequency point where the magnitude/phase error exhibits its maximum value is added to the initial set of frequency points and one additional SBG step is performed on it. Each node contraction and branch removal is performed only if the error specifications are met at all frequency points of the set. This process is iteratively applied until the required accuracy is ensured for the complete frequency range defined by the user.

#### 4. Simplification during generation techniques

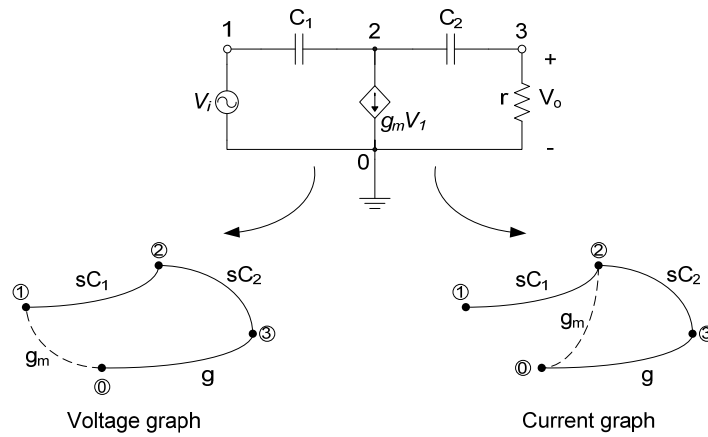
Simplification During Generation (SDG) techniques start from some formulation of the network equations of a circuit in the form of a graph or matrix (e.g. voltage and current graphs, directed graphs, etc.) and solve them by directly generating the most simplified expression that fulfils some error criteria. Simplification During Generation approaches have two main advantages: first, the analysis is faster, as no time is wasted generating terms that would be neglected if a SAG technique with analogous error specifications were applied on the exact symbolic results. Second, its smaller memory consumption enables the analysis of much larger circuits without exhausting the computer resources.

Typically, SDG algorithms generate symbolic terms in decreasing order of magnitude until the number of terms is enough to model the behavior of the circuit with a given accuracy. Two tasks are involved in this problem: (a) a term generator, i.e., an algorithm that can generate symbolic terms in decreasing order of magnitude, according to their influence on the circuit behavior; and (b) an error control mechanism, that can determine when the generated symbolic expression fulfils the error specifications.

## 4.1. Term generation

### 4.1.1. Two-graph approach

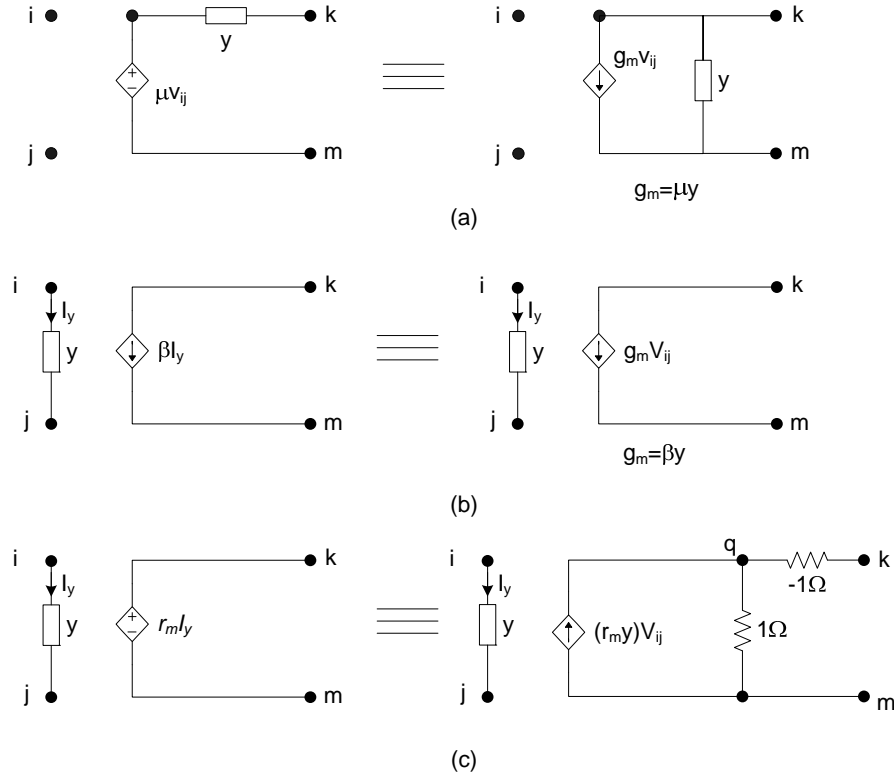
The first reliable algorithms capable of efficiently generating terms in decreasing order of magnitude for large circuits were reported in [20]–[23]. All these algorithms are based on the two-graph method. This method uses two graphs: a voltage graph and a current graph. Given the small-signal model of a circuit, they can be easily built. Each circuit node corresponds to an equivalent node in the voltage and current graphs. Each passive device corresponds to one branch between the graphs nodes that correspond to its terminal nodes. For voltage-controlled current sources, the voltage graph is constructed with the controlled branch and the current graph with the controlling branch. Fig. 4 shows a sample circuit and the corresponding voltage and current graphs. All other types of controlled sources are converted to the interconnection of passive elements and voltage-controlled current sources, as illustrated in Fig. 5 [24]. For convenience, inductors are usually replaced by the interconnection of capacitors and voltage-controlled current sources. Each branch is assigned the admittance (or transadmittance) of the corresponding circuit element.



**Figure 4:** A simple circuit and its corresponding voltage and current graphs.

The computation of the most significant terms for each power of the frequency  $s^k$ , in the numerator and denominator of the network function, can be expressed as the following graph problem: “Given the voltage and the current graphs of a circuit with  $n$  nodes and  $b$  branches, enumerate subsets of branches in decreasing order of magnitude that:

- (1) constitute a spanning tree in the voltage graph VG;
- (2) constitute a spanning tree in the current graph CG;



**Figure 5:** Equivalent circuits for (a) voltage-controlled voltage sources (VCVS), (b) current-controlled current sources (CCCS) and (c) current-controlled voltage sources (CCVS).

(3) contain  $k$  capacitance branches and  $(n - k - 1)$  (trans)conductance branches.”

A valid symbolic term is given by the product of all branch admittances included in each common spanning tree of the voltage and current graphs. The sign of the term is determined separately, using the topological information of both graphs [25]. Although the sign of each term is difficult to obtain by hand-made calculations, it can be very efficiently calculated with a negligible computational time.

The procedure behind the methodologies in [12],[20]-[23],[26],[27] is the following: first, select one of the graphs, generate spanning trees with  $k$  capacitance branches in decreasing order of magnitude in that graph; then, for each tree, check if it is also a spanning tree in the current graph. The generation of spanning trees in decreasing order follows the algorithm in [28], originally proposed for graphs with a single type of branches but extensible to graphs with two types of branches [26].

#### 4.1.2. Matroid intersection approach

The term generation problem can be also formulated in terms of the matroid theory [29]. A *matroid*  $M = (E, \mathfrak{I})$  is a structure in which  $E$  is a finite set of elements and  $\mathfrak{I}$  is a family of subsets of  $E$ , which satisfy some axioms [29]. A subset  $I$  in  $\mathfrak{I}$  is an *independent set* of the matroid  $M = (E, \mathfrak{I})$ . A maximal independent set is a *base* of

the matroid. A matroid  $M = (E, \mathfrak{I})$  is the *graphic matroid* of graph  $G$  if  $E$  is the set of branches of  $G$  and a subset  $I \subseteq E$  is in  $\mathfrak{I}$  if and only if  $I$  is a cycle-free subset of branches. In a connected graph, a base of its graphic matroid corresponds to a spanning tree.

Let  $\pi$  be a partition that separates  $E$  into  $m$  disjoint blocks  $B_1, \dots, B_m$ , and let  $d_i$  ( $i=1, \dots, m$ ) be  $m$  non-negative integers. Then,  $M = (E, \mathfrak{I})$  is a *partition matroid* if  $\mathfrak{I}$  is the family of subsets  $I$  that satisfy  $|I \cap B_i| \leq d_i$  ( $i=1, \dots, m$ ).

The term generation implies the ranked (in decreasing order of weight) enumeration of bases that are common to three matroids [29], commonly known as a weighted three-matroid intersection problem. For instance, the three matroids involved in the formulation of the tree enumeration problem in the two-graph method described above are: two graphic matroids associated to the voltage and current graphs where each spanning tree is a base of the matroid, and a partition matroid where each set of  $n-1$  branches with  $k$  capacitances is a base of the matroid.

The formulation of the term generation problem in terms of matroid theory was first proposed in [30], only for the two-graph method. Later on, a formulation and comparison, among the two-graph method, the directed-tree enumeration method, the Coates flow graph method, the Laplace determinant expansion and the parameter extraction method of Sannuti and Puri was introduced in [31],[32].

The intersection of three matroids is considered to be, in general, a NP-hard (as hard as any non-deterministic polynomial time) problem. If a problem can be formulated as a weighted 2-matroid intersection problem, then a polynomial time algorithm exists, being that of Camerini and Hamacher [33] one of the most efficient ones. Moreover, the efficiency can be further improved by exploiting specific properties of the matroids under consideration.

The analysis in [31],[32] is extremely interesting to select the optimum formulation method when the matroid intersection theory is applied. This is illustrated in Table 1 for the examples depicted in Fig. 6 [31],[32]. Four different circuits have been analyzed using the different analysis methods. The second and third rows show the number of nodes and devices (after replacement of semiconductor devices by small-signal models) in those circuits. This gives a clear idea of the problem complexity. The fourth row shows the total number of spanning trees in the voltage graph for each case. The fifth row shows the number of spanning trees in the current graph (normalized to the values in row three). Then, the sixth row shows a lower bound on the number of common spanning trees to both graphs. Finally, the remaining rows show (also normalized to the results in row three) the calculation of the number of terms generated with the different approaches.

**Table 1: Number of Terms with Different Analysis Techniques for the Circuits of Fig. 6.**

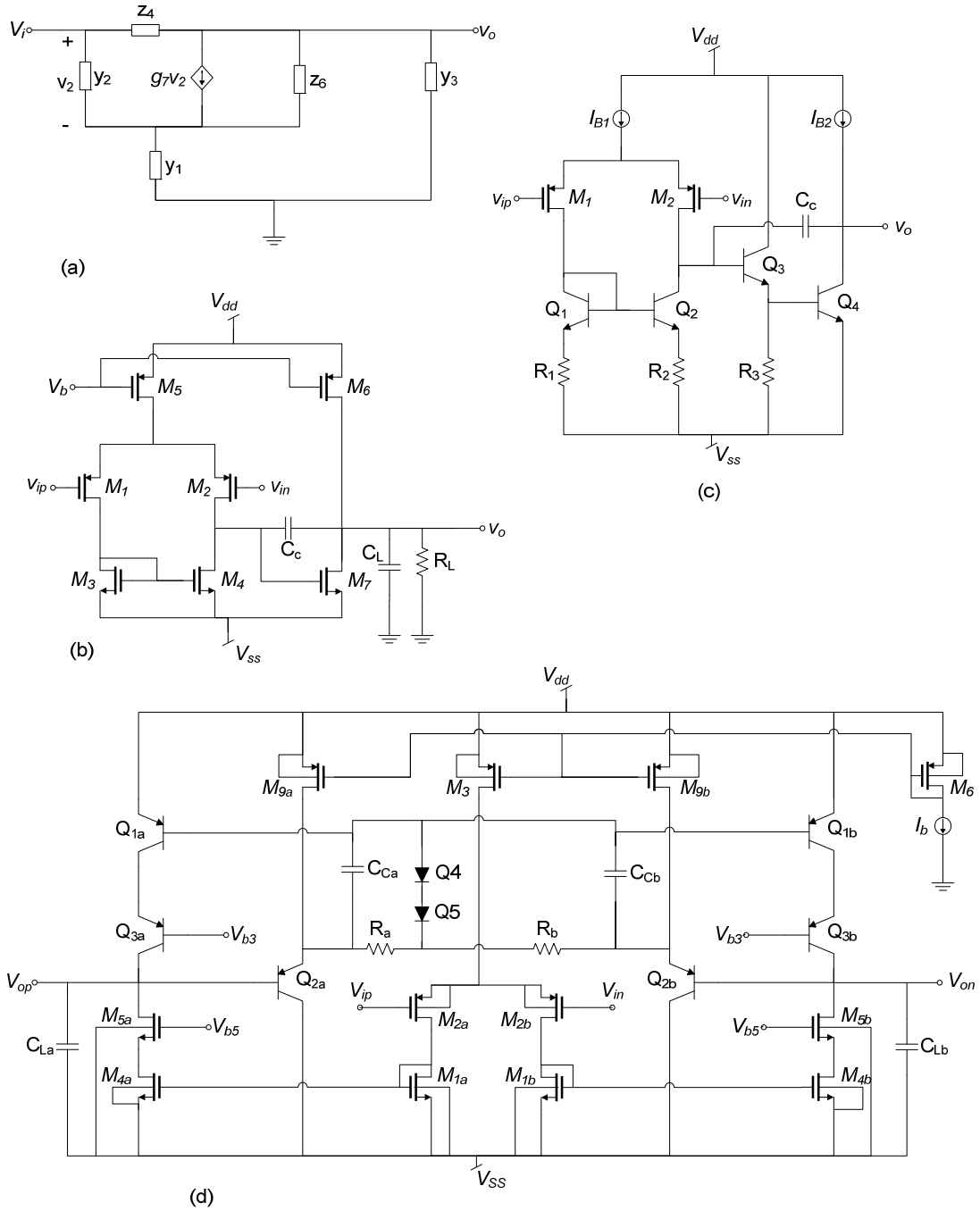
	Fig. 6(a)	Fig. 6(b)	Fig. 6(c)	Fig. 6(d)
# nodes	4	5	9	16
# circuit elements	8	16	31	70
# spanning trees in $G_V$	36	346	$3.55 \cdot 10^5$	$2.96 \cdot 10^{11}$
# spanning trees in $G_I$	1	2.13	1.01	2.20
lower bound on # common spanning trees	0.667	0.694	0.553	0.222
exact # different terms	0.722	0.694	0.553	
# terms with directed-tree enumeration	2.08	1.43	1.18	11.9
# terms with signal flow graph	1.92	4.17		
# terms with Coates flow graph / Laplace determinant expansion	2.85	1.20	6.23	5.197
# terms with parameter extraction of Sannuti & Puri	1.22	8.62	195	$2.55 \cdot 10^7$

Assuming that an algorithm with the same efficiency is available to enumerate bases in the matroid intersection problem corresponding to each of the methods in Table 1, it can be concluded that the number of bases in the two-graph method is smaller than in the other methods. This means that many bases are generated in other methods that do not correspond to a valid symbolic term in the final symbolic expressions. This ratio tends to increase when the circuit complexity increases. It is also interesting that the number of bases in the voltage graph is smaller than in the current graph.

Moreover, not only the number of bases is smaller, but algorithms that exploit the specific circumstances involved in the problem and achieve a better efficiency than the standard algorithm in [33] have been reported for the two-graph approach.

Therefore, the two-graph method is the best formulation method for an SDG implementation. As the enumeration of common spanning trees for a given coefficient of the network function is a 3-matroid intersection problem and the matroid intersection algorithm in [33] is a 2-matroid intersection algorithm, three possibilities arise:

- a) *Intersect the voltage graphic matroid and the partition matroid and for each enumerated base check if it is also a base in the current graphic matroid.* By exploiting specific features of the two matroids involved, the computational complexity of this procedure can be considerably lowered. This approach has been implemented in [27],[30].



**Figure 6:** (a) Simple network, (b) CMOS Miller OTA, (c) BiCMOS Miller OTA and (d) BiCMOS fully differential OTA.

This approach has in its efficiency for large circuits its main drawback. This is due to the fact that only a part of the spanning trees generated (common to VG and the partition matroid) will be spanning trees of the current graph. What is more, as the size of the circuit increases, the number of non-valid terms grows fast. Therefore, when the generation process is started, a lot of resources are wasted in the generation of terms that will not be valid spanning trees of the

CG.

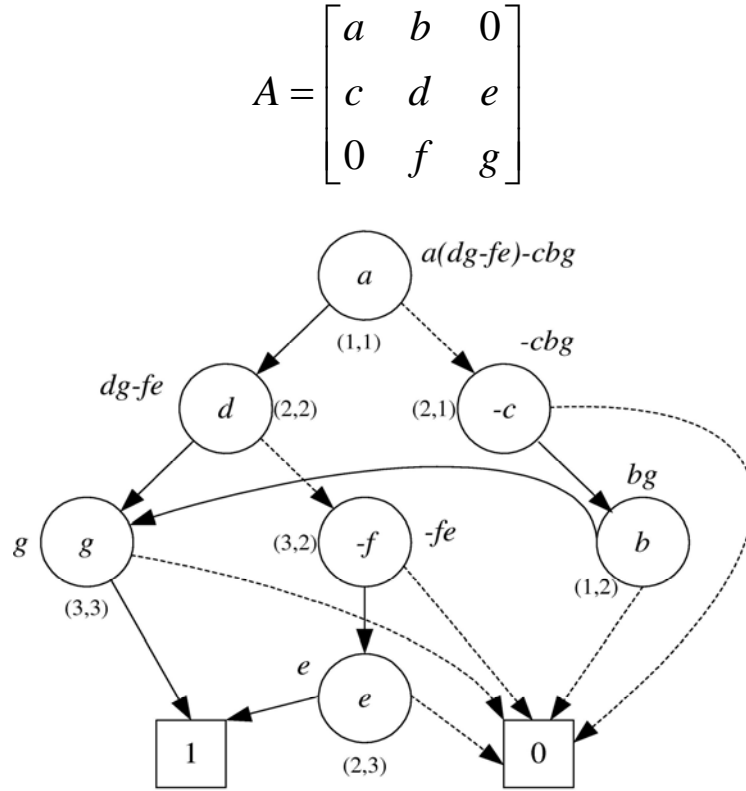
- b) *Intersect the current graphic matroid and the partition matroid and for each enumerated base check if it is also a base in the voltage graphic matroid.* This is the dual approach to the previous one. However, as demonstrated with the examples in Table 1, the number of bases is larger in the current graph, and, therefore, this alternative is disadvantageous with respect to the previous one and will not be paid any more attention.
- c) *Intersect the two graphic matroids and, for each enumerated base, check if it contains the desired number of capacitances.* Common spanning trees of the voltage and current graphs in decreasing order of tree admittance product are generated using for instance the algorithm in [33]. The computational complexity is larger than in the first approach above but, in this case, no time is wasted in generating spanning trees of one of these graphs which, afterwards, are not spanning trees in the other graph. However, no control is provided on the branch type (frequency-dependent or non-frequency dependent elements), and, hence, the weight of capacitor admittances must be evaluated at a fixed frequency.

The advantage of this method is that all the generated terms are valid. However, there exists an important drawback: many terms can be generated more than once (when they are generated at several frequency samples) and this means a loss of efficiency. A definitive answer on the advantages with respect to the first approach above (intersection of voltage graphic matroid and partition matroid) depends on the number of frequency samples, but this is related to the error control methodology that is discussed in section 4.2.

#### 4.1.3. Determinant decision diagram (DDD)-based approach

In this section, we describe another approach that is based on the determinant decision diagram (DDD) concept [34],[35]. DDDs are compact graphs that have been used to represent matrix determinants and they are constructed for the Laplace expansion method. Each coefficient of the admittance matrix is considered as one distinct symbol and each of them is represented into DDDs as a non-terminal vertex along with its sign and labeled as  $a_i$  ( $a$  to  $g$  in Fig. 7) [34]. Further, a DDD has two terminal vertices, namely: *0-terminal* vertex and *1-terminal* vertex, and one starting vertex, called *root* vertex. Each non-terminal vertex has two edges: *1-edge* (solid arrow) and *0-edge* (dashed arrow), and a path from the root vertex to the 1-terminal is called *1-path*, as depicted in Fig. 7.

In order to build the DDD from the matrix  $A$ , all coefficients must be sorted in descending form, for instance:  $a > c > b > d > f > e > g$ . A heuristic method of vertex sorting is given in [34]. Then, each vertex represents a determinant matrix  $D$  defined recursively as:

**Figure 7:** DDD of the matrix A.

1.  $D = 1$ , if the vertex is pointing out to the  $1$ -terminal.
2.  $D = 0$ , if the vertex is pointing out to the  $0$ -terminal.
3.  $D = a_i s(a_i) D_{a_i} + D_{a_i}^-$ , for each non-terminal vertex. Here  $D_{a_i}$  and  $D_{a_i}^-$  are the determinants of the vertices that point out to the  $1$ -edge and  $0$ -edge, respectively. The sign of each vertex  $a_i$  in any  $1$ -path is defined as:

$$s(v) = \prod \text{sign}(r(x) - r(v)) \text{sign}(c(x) - c(v)) \quad (33)$$

where  $r(x)$  and  $c(x)$  are the indexes of the row and column associated to each coefficient of matrix A. The  $\text{sign}$  function is defined as:

$$\text{sign}(u) = \begin{cases} 1 & \text{for } u > 0 \\ -1 & \text{for } u < 0 \end{cases} \quad (34)$$

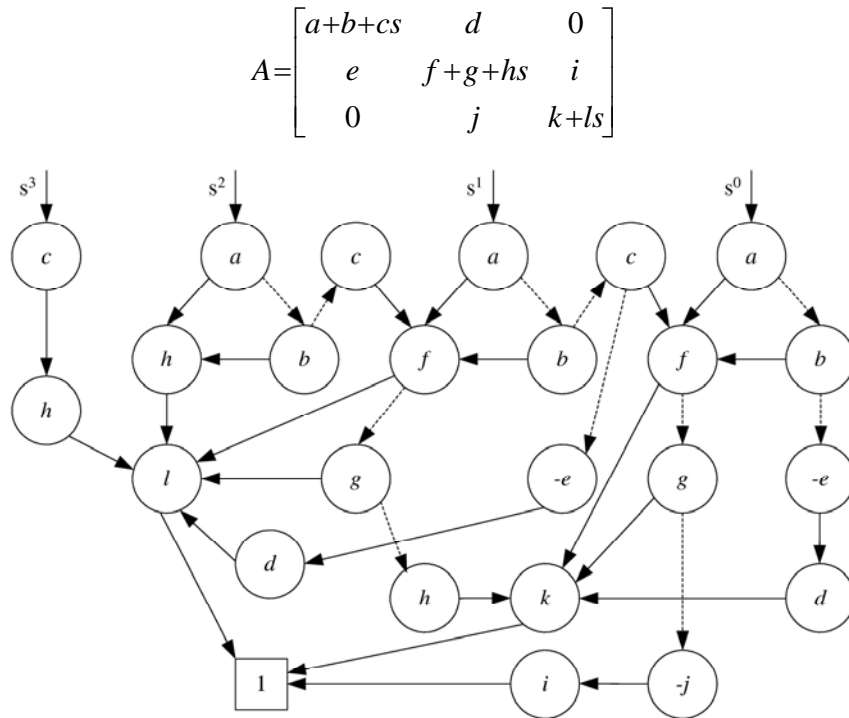
For the case that  $v$  has an edge pointing to the  $1$ -terminal vertex, then  $s(v) = 1$ . For instance, let us consider the vertex  $c$  with its indexes given by:  $r(v) = 2$  and  $c(v) = 1$ . The indexes of the following vertex in the  $1$ -edge, labeled as  $b$  are given by:  $r(x) = 1$  and  $c(x) = 2$ . By applying (33) and (34), the negative sign of the vertex  $c$  is obtained.



On the other hand, a 1-path in a DDD is defined as a path from root vertex to the 1-terminal, considering all symbolic terms along with the sign of the vertices that originate all the 1-edges. Each 1-path represents a product of terms represented by a 1-edge whereas the addition operation is represented by a 0-edge. For instance, one can see that there are three product terms in Fig. 7, given by:  $-adg$ ,  $-aef$ , and  $-bcg$ ; they are the product terms of the determinant of the matrix  $A$ .

Despite the fact that the determinant of a matrix can efficiently be represented by DDDs, it is still necessary, however, to expand each vertex in the DDD, because each coefficient in the admittance matrix is often an  $s$  polynomial composed with different symbols. In [35], the generation of symbolic transfer functions in the form of polynomials, as shown by (3), has been introduced. This approach called  $s$ -expanded DDDs is based upon multiple-root DDDs in order to represent the determinant and its cofactors. Let us substitute each coefficient of the matrix  $A$ , shown in Fig. 7, by an  $s$  polynomial. Thus, the matrix  $A$  is rewritten as shown in Fig. 8.

Therefore, each vertex of the DDD, shown in Fig. 7, must first be sorted, expanded and, then, the  $s$ -variables extracted [35]. Thus, each power of  $s$  becomes a root of the multiple-root DDDs, as shown in Fig. 8. This technique expands all  $s$  polynomials of the original determinant in a DDD and therefore, symbolic transfer functions can be generated by finding all the  $1$ -paths for each  $s$ -root.



**Figure 8:**  $S$ -expanded DDD of the matrix  $A$ .

As has been reported in [35], a disadvantage of DDDs and  $s$ -expanded DDDs methods, that are based on the Laplace expansion method, is that the original matrix

suffers from term cancellations. Thus, for each vertex generated in the  $s$ -expanded DDDs method, cancelling terms are indirectly introduced. In order to improve the generation of dominant terms, cancelling terms should be removed. From the admittance matrix  $A$ , one can see that there are patterns related with cancelling terms, as shown in Fig. 9.

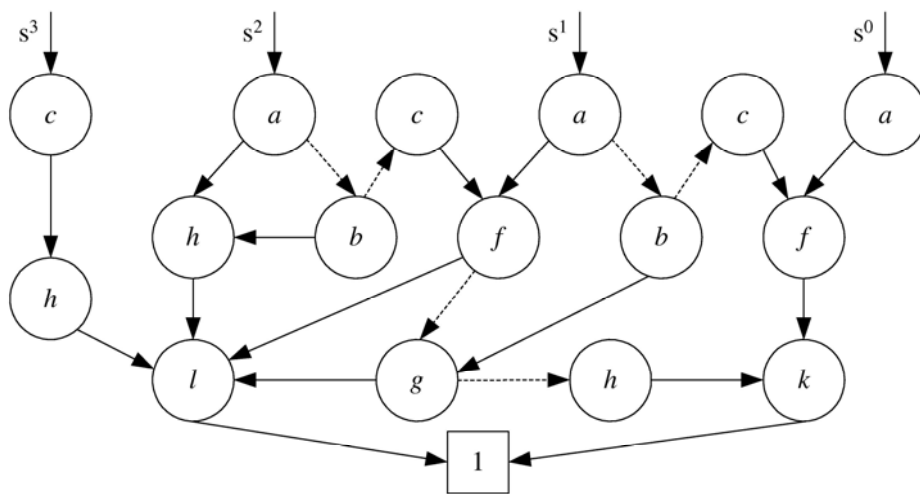
An efficient algorithm to remove the patterns shown in Fig. 9 has been introduced in [36]. In this manner, the  $s$ -expanded DDD shown in Fig. 8 can be reduced as depicted in Fig. 10.

$$\begin{array}{ccc} \begin{array}{c} l \quad k \\ i \begin{bmatrix} p & -p \\ -p & p \end{bmatrix} \\ j \end{array} & \begin{array}{c} l \quad k \\ i \begin{bmatrix} p & -q \\ -p & q \end{bmatrix} \\ j \end{array} & \begin{array}{c} l \quad k \\ i \begin{bmatrix} p & -p \\ -q & q \end{bmatrix} \\ j \end{array} \end{array}$$

**Figure 9:** Patterns that generate cancelling terms in the admittance matrix

In order to generate only the dominant terms by using DDDs, several algorithms based on dynamic programming (DP) have been proposed [37]–[39]. The main idea is to compute partial dominant terms for each DDD vertex pointing to a  $l$ -edge. Let us suppose that there are  $n$  vertices in a path, from root vertex to the  $l$ -terminal. Each vertex is composed of an  $s$  polynomial; therefore, each coefficient is first checked in order to avoid generating terms more than twice. If it does not exist, then the largest term is computed and stored in a term-list by visiting all the DDD vertices linked to  $0$ -edge, where its numerical value also is computed.

However, the method described above is not applicable to  $s$ -expanded DDDs, because there are vertices with incoming  $l$ -edges and  $0$ -edges. It is worth mentioning that it is necessary to duplicate some vertices in order to apply the method described above, as has been described in [37]–[39]. To solve this drawback, another DP-based algorithm



**Figure 10:**  $S$ -expanded multiroot DDD without cancelling terms.

has also been proposed in [39]. This algorithm is based on finding the shortest path in edge-weighted DDDs graphs by using the following edge weight:

1.  $0$ -edge with weight  $0$ .
2.  $1$ -edge with weight  $= -\log|a_i|$

Here  $|a_i|$  is the numerical value of the DDD vertex  $a_i$  of the  $1$ -edge. As a consequence the total weight of a path is given as:

$$w = -(\log(a_1) + \log(a_2) + \dots \log(a_i)) \quad (35)$$

Once a shortest path has been found in a DDD, it is subtracted in a similar form as  $s$ -variables [35] and, then, the next shortest path is searched. A disadvantage of this approach is that one needs to visit each vertex in a DDD graph for each shortest path. In response to this drawback, a reverse DDD graph-based new algorithm has also been introduced in [39]. Basically, the root vertex, the two terminal vertices and all edges along with their directions in a normal DDD graph are reversed. Therefore, the two terminal vertices are now the root vertices and the root vertex in DDDs becomes the new terminal vertex. In the same way as normal DDD graphs,  $1$ -paths and path weights are defined for reverse DDD graphs. Thus, the dominant terms correspond to the shortest paths in the corresponding reverse DDD graph and they are extracted in the same way as  $s$ -variables [35]. Finally, after the first shortest path has been found, the next shortest path can be found by only visiting the newly added vertices created by the subtraction operation. From the DP algorithms described above, we can conclude that the latter algorithm is more efficient in order to generate dominant terms based on DDDs, since it can be applied to any topology of a circuit, which can be represented by a normal DDD or  $s$ -expanded DDD graphs.

## 4.2. Error control

Term generation must be accompanied by an appropriate error control methodology. Moreover, the error control methodology determines to a certain extent the appropriate term generation approach among the alternatives discussed above.

### 4.2.1. Coefficient-based approach

The basic error control methodology is a simple extension of the methods used in simplification after generation techniques discussed in Section 2. Approaches in [20]-[23],[26],[27] generate the  $P$  most significant terms in (3) for each network function coefficient,  $h_k(\mathbf{x})$ , until the sum of the generated terms represents a given fraction of the total magnitude of the coefficient:

$$\left| h_k(\mathbf{x}_o) - \sum_{l=1}^P h_{kl}(\mathbf{x}_o) \right| < \varepsilon_k |h_k(\mathbf{x}_o)| \quad (36)$$

where  $h_k(\bullet)$  represents any coefficient from the network function numerator or denominator,  $\mathbf{x}_o$  represents a design point of the circuit parameters and  $\varepsilon_k$  is a threshold error.

As shown in (36), the total magnitude of each circuit coefficient,  $h_k(\mathbf{x}_o)$ , must be known a priori. The same numerical interpolation technique used in the second SBG error control mechanism in Section 3.5 can be applied here.

This error control methodology is naturally associated to the first term generation methodology. If each coefficient in the network function were generated with the same threshold error, there would be no magnitude and phase deviations. However, even if the same error threshold were specified for each coefficient, the real error would be different for each one due to the discrete characteristic of the simplification process. As that real error is unpredictable a priori, the same happens for the magnitude and phase deviations, which may become really large.

A possible solution is to use (31) to back-propagate magnitude and phase errors specified by the user to individual errors of the network function coefficients. However, such back-propagation is extremely conservative and many more terms than those strictly necessary to meet the magnitude and phase error constraints are generated.

#### **4.2.2. Sensitivity-based approach**

This procedure starts with the elimination of unimportant coefficients [40]. This elimination is performed according to a ranking function that orders the contributions of the coefficients and eliminates those having the lowest contribution while the error constraints are met. To avoid the generation of too many terms in the remaining coefficients, only a part of the total error allowed by the user is applied in this step.

In a second step, the largest term of each of the remaining coefficients is generated and added to the final symbolic expression. Then, having queued the following largest terms of each coefficient, a sensitivity analysis allows deciding which coefficient is contributing the most to the circuit behavior. The largest term of this coefficient is added to the symbolic expression and the error constraints are checked. If the specifications are fulfilled, the term generation is finished; otherwise, the sensitivity analysis is again used to add a new term to the final symbolic expression. This procedure is repeated until the error specifications given by the user are met.

Using this methodology, the generated symbolic expression will be more compact than in the previous case. However, as in the first approach, to control the accuracy of the results, a dense set of frequency samples is needed to compare the evaluation of the generated expression to the exact circuit behavior. Then, the accuracy is not guaranteed between each pair of frequency samples.

For the generation of terms in decreasing order of magnitude within each coefficient two possibilities arise:

- *Use the intersection of the voltage graphic matroid and the partition matroid.* If this approach is used, the generated terms have to be checked to see if they are also valid spanning trees of the current graph. Unfortunately, for circuits with many voltage-controlled current sources (i.e. those circuits composed of transistors), the number of terms that are valid for the intersection problem but are not spanning trees of the current graph grows exponentially with the circuit size. Therefore, a lot of terms will be needed to generate a valid term for the symbolic expression.
- *Use of the intersection of the two graphic matroids.* In this case, the enumeration of bases is more costly but it is guaranteed that all the generated ones are valid. Bases can be generated only a single frequency values. In the approach in [41],[42], the sensitivity analysis is used to find the frequency at which each coefficient dominates the circuit behavior. At this frequency value, the generated terms in decreasing order of magnitude are expected to belong to the desired coefficient. However, except for low-order coefficients, these frequency values are usually too close, so terms belonging to adjacent coefficients are generated in the process. Of course, all the terms that do not belong to the desired coefficient have to be neglected. This, again, imposes an extra overhead on the number of generated terms to obtain a valid set of spanning trees.

#### 4.2.3. Sampling-based approach

In [43], a set of frequency samples belonging to the frequency range of interest is used. For each of these samples, the matroid intersection algorithm involving the two graphic matroids, is executed, until the error criteria at such frequency value is fulfilled according to a relative threshold defined by the user. Joining together all the generated terms at all the frequency samples provides a symbolic expression valid at such samples.

A major disadvantage of this approach is that, pasting together all the terms generated at each frequency sample implies that the final symbolic expression will have more terms than those strictly necessary to meet the error constraints. And, the larger the number of frequency samples, the larger the number of extra (unnecessary) terms.

Also, the accuracy is not guaranteed between each pair of frequency samples. This will be guaranteed only in case that an infinite number of frequency samples are used; but this is obviously impossible. The possibility of violating the error constraints decreases (although never reaching zero) if the number of frequency samples is increased, but this also implies higher CPU time and more conservative results.

Another approach that palliates the problems of previous approaches is reported in [44]. To ensure that valid terms are always generated without needing a large number of frequency samples, thus, ensuring that the resulting symbolic expression is as compact as possible, this methodology proceeds as follows. For one frequency sample, and within the range defined by the user, the term generation based on the intersection of the two graphic matroids is executed until the errors at that frequency are within the specifications. By doing so, it is only guaranteed that the generated

expression fulfils the error constraints in the selected frequency value. Thus, we need some mechanism to find out if, within the frequency range, there are some frequency values in which error constraints are exceeded. This test is performed by the following maximum error detection algorithm. The difference between the magnitude and phase behaviors of the original circuit and those of the evaluation of the generated expression is calculated. Then, the maximum for the magnitude and phase errors are obtained by using the same methodology than in Section 3.5 and if any of these two errors are beyond the specifications, the frequency value at which the maximum of the errors is achieved is added to the existing set of frequency samples.

This procedure is performed iteratively until the error specifications are met in the complete frequency range defined by the user. A small number of iterations are usually enough.

This algorithm allows the fast generation of symbolic expressions with full accuracy. However, the accumulation of different frequency sampling points may result in a symbolic expression whose complexity is larger than strictly needed. This is because, as a consequence of the generation of symbolic terms in an additional frequency point, some terms already generated at other frequency may become obsolete. Note that this problem has already been drastically minimized by using a small number of frequency samples. However, to ensure that the complexity of the results is minimal, a post-processing step after the simplification process is applied. All the symbolic terms, generated at the different frequency samples, are queued according to their weight within each coefficient. Then, taking into account the contribution of the different coefficients in the network function to the circuit behavior, symbolic terms are iteratively neglected, starting with those having the smallest weight, while the error specifications are not exceeded.

## 5. Conclusion

The exponential growth of symbolic analysis results with the circuit size demands the application of approximated analysis techniques. Historically, simplification after generation techniques, i.e., techniques that approximate the symbolic expressions once they have been generated, were the first ones to appear. They palliated the interpretation problem but the limitation on the analyzable circuit size still remained. This has been solved by the introduction of simplification before generation techniques, which simplify the system of circuit equations, and simplification during generation techniques, which directly generate simplified symbolic expressions. Most efficient analysis techniques are based on the two-graph method, which more recently have been modeled as matroid intersection problems, and on determinant decision diagrams. Approximated analysis techniques must include a suitable error control mechanism, which decides the appropriate approximate symbolic results for a given accuracy.

## Acknowledgement

The preparation of this chapter has been partially supported by the TIC-2532 Project, funded by Consejería de Innovación, Ciencia y Empresa, Junta de Andalucía and by

the Project TEC2010-14825/MIC, funded by the Spanish Ministry of Science and Innovation with support from ERDF. Dr. C. Sánchez-López thanks the support of the JAE-Doc program of CSIC, co-funded by FSE.

## References

- [1] G. Wierzba, *Sspice user manual*, Version 1.0. Michigan State University, East Lansing, February 1991.
- [2] G. Gielen and W. Sansen, *Symbolic analysis for automated design of analog integrated circuits*. Boston: Kluwer Academic, 1991.
- [3] F. V. Fernández, A. Rodríguez-Vázquez, and J. L. Huertas, “Interactive AC modeling and characterization of analog circuits via symbolic analysis,” *Analog Integrated Circuit and Signal Processing*, vol. 1, pp. 183–208, November 1991.
- [4] A. Liberatore, A. Luchetta, S. Manetti and M. C. Piccirilli, “A new symbolic program package for the interactive design of analog circuits” in *IEEE International Symposium on Circuits and Systems*, 1995, vol. 3, pp. 2209-2212.
- [5] G. Gielen, H. Walscharts, and W. Sansen, “ISAAC: A symbolic simulator for analog integrated circuits,” *IEEE Journal of Solid-State Circuits*, vol. 24, pp. 1587–1597, December 1989.
- [6] F. V. Fernández, A. Rodríguez-Vázquez, J. D. Martín, and J. L. Huertas, “Formula approximation for flat and hierarchical symbolic analysis,” *Analog Integrated Circuits and Signal Processing*, vol. 3, no. 1, pp. 43–58, Kluwer, January 1993.
- [7] R. E. Moore, *Methods and applications of interval analysis*. Studies in Applied Mathematics, Philadelphia, 1979.
- [8] J.-J. Hsu and C. Sechen, “Fully symbolic analysis of large analog integrated circuits”, in *IEEE Custom Integrated Circuits Conference*, 1994, pp. 457-460.
- [9] J. J. Hsu and C. Sechen, “DC small signal symbolic analysis of large analog integrated circuits,” *IEEE Transactions Circuits and Systems I*, vol. 41, no. 12, pp. 817-828, December 1994.
- [10] J. J. Hsu and C. Sechen, “Low-frequency symbolic analysis of large analog integrated circuits” in *IEEE Custom Integrated Circuits Conference*, 1993, pp. 14.7.1-14.7.5.
- [11] R. Sommer, E. Hennig, G. Droge and E. H. Horneber, “Equation-based symbolic approximation by matrix reduction with quantitative error prediction,” *Alta Frequenza*, vol. 5, no. 6, pp. 317-325, November 1993.



- [12] Q. Yu and C. Sechen, "A unified approach to the approximate symbolic analysis of large analog integrated circuits," *IEEE Transactions on Circuits and Systems I*, vol. 43, no. 8, pp. 656-669, August 1996.
- [13] W. Daems, G. Gielen and W. Sansen, "Circuit complexity reduction for symbolic analysis of analog integrated circuits", in *IEEE/ACM Design Automation Conference*, 1999, pp. 958-963.
- [14] W. Daems, G. Gielen and W. Sansen, "Circuit simplification for the symbolic analysis of analog integrated circuits", *IEEE Transactions on Computer-Aided Design*, vol. 21, no. 4, pp. 395-407, April 2002.
- [15] O. Guerra, J.D. Rodríguez-García, E. Roca, F.V. Fernández and A. Rodríguez-Vázquez, "A simplification before and during generation methodology for symbolic large-circuit analysis", in *IEEE International Conference on Electronics, Circuits and Systems*, 1998, vol. 3, pp. 81-84.
- [16] O. Guerra, J. D. Rodríguez-García, E. Roca, F. V. Fernández and A. Rodríguez-Vázquez, "An accurate error control mechanism for simplification before generation algorithms", in *IEEE Design, Automation and Test in Europe Conference*, 1999, pp. 412-416.
- [17] J. D. Rodríguez-García, O. Guerra, E. Roca, F. V. Fernández and A. Rodríguez-Vázquez, "Error control in simplification before generation algorithms for symbolic analysis of large analogue circuits," *Electronic Letters*, vol. 35, no. 4, pp. 260-261, February, 1999.
- [18] J. Vlach and K. Singhal, *Computer methods for circuit analysis and design*. Van Nostrand Reinhold, 1994.
- [19] F. V. Fernández, O. Guerra, J. D. Rodríguez-García and A. Rodríguez-Vázquez, "Symbolic analysis of analog integrated circuits: the numerical reference generation problem," *IEEE Transactions on Circuits and Systems-II*, vol. 45, no. 10, pp. 1351-1361, October 1998.
- [20] F. V. Fernández, P. Wambacq, G. Gielen, A. Rodríguez-Vázquez and W. Sansen, "Symbolic analysis of large analog integrated circuits by approximation during expression generation," in *IEEE International Symposium on Circuits and Systems*, 1994, vol. CAD, pp. 25-28.
- [21] P. Wambacq, F.V. Fernández, G. Gielen and W. Sansen: "Efficient symbolic computation of approximated small-signal characteristics", in *IEEE Custom Integrated Circuits Conference*, 1994, pp. 461-464.
- [22] Q. Yu and C. Sechen, "Generation of color-constrained spanning trees with application in symbolic circuit analysis," in *Fourth Great Lakes Symposium on VLSI*, 1994, pp. 252-255.



- [23] Q. Yu and C. Sechen, "Approximate symbolic analysis of large analog integrated circuits," in *IEEE International Conference on Computer-Aided Design*, 1994, pp. 664-671.
- [24] P. M. Lin, *Symbolic network analysis*. Amsterdam: Elsevier, 1991.
- [25] S.P. Chan, *Introductory Topological analysis of electrical networks*. Holt, Rinehart and Winston, 1969.
- [26] P. Wambacq, F. V. Fernández, G. Gielen, W. Sansen and A. Rodríguez-Vázquez, "Algorithm for efficient symbolic analysis of large analogue circuits," *Electronics Letters*, pp. 1108-1109, July 1994.
- [27] P. Wambacq, F. V. Fernández, G. Gielen, W. Sansen and A. Rodríguez-Vázquez, "Efficient symbolic computation of approximated small-signal characteristics of analog integrated circuits," *IEEE Journal of Solid-State Circuits*, vol. 30, no. 3, pp. 327-330, March 1995.
- [28] H.N. Gabow, "Two algorithms for generating weighted spanning trees in order", *SIAM Journal of Computing*, vol. 6, no. 1, pp. 139-150, March 1977.
- [29] E.L. Lawler, *Combinatorial optimization: networks and matroids*. Holt, Rinehart and Winston, 1976.
- [30] Q. Yu and C. Sechen, "Efficient approximation of symbolic network functions using matroid intersection algorithms," in *IEEE International Symposium on Circuits and Systems*, 1995, pp. 2088-2091.
- [31] P. Wambacq, F. V. Fernández, G. Gielen, W. Sansen and A. Rodríguez-Vázquez, "A family of matroid intersection algorithms for the computation of approximated symbolic network functions" in *IEEE International Symposium on Circuits and Systems*, 1996, pp. 806-809.
- [32] P. Wambacq, "Symbolic analysis of large and weakly non-linear analog integrated circuits" Ph. D. Thesis, KU Leuven, 1996.
- [33] P. M. Camerini and H.W. Hamacher, "Intersection of two matroids: (condensed) border graph and ranking", *SIAM Journal Discrete Mathematics*, vol. 2, pp. 16-27, February 1989.
- [34] C.-J. Shi and X.-D. Tan, "Canonical symbolic analysis of large analog circuits with determinant decision diagrams", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 19, pp. 1-18, January 2000.
- [35] C.-J. Shi and X.-D. Tan, "Compact representation and efficient generation of s-expanded symbolic network functions for computer-aided analog circuit design", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, pp. 813-827, July 2001.

- [36] S. X.-D. Tan and C.-J. Shi, "Parametric analog behavioral modeling based on cancellation-free DDDs", in *Proc. IEEE International Workshop on Behavioral Modeling and Simulation*, 2002, pp. 25-31.
- [37] W. Verhaegen and G. Gielen, "Symbolic determinant decision diagrams and their use for symbolic modeling of linear analog integrated circuits", *International Journal of Analog Integrated Circuits and Signal Processing*, vol. 31, pp. 119-130, 2002.
- [38] W. Verhaegen and G. Gielen, "Efficient DDD-based symbolic analysis of linear analog circuits", *IEEE Transactions on Circuits and Systems II: Analog and digital signal processing*, vol. 49, pp. 474-487, July 2002.
- [39] X.-D. Tan and C.-J. Shi, "Efficient approximation of symbolic expressions for analog behavioral modeling and analysis", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and System*, vol. 23, pp. 907-918, June 2004.
- [40] W. Daems, W. Verhaegen, P. Wambacq, G. Gielen and W. Sansen, "Evaluation of error control strategies for the linear symbolic analysis of analog integrated circuits", *IEEE Transactions Circuits and Systems I*, vol. 46, no. 5, pp. 594-606, May 1999.
- [41] P. Dobrovolny, P. Wambacq, G. Gielen and W. Sansen, "Efficient symbolic analysis of large analog circuits using sensitivity-driven ranking of matroid intersections," in *IEEE International Symposium on Circuits and Systems*, 1998, vol. 6, pp. 29-32.
- [42] P. Wambacq, P. Dobrovolny, G. Gielen and W. Sansen, "Symbolic analysis of large analog circuits using a sensitivity-driven enumeration of common spanning trees," *IEEE Transactions Circuits and Systems II*, vol. 45, no. 10, pp. 1342-1350, Oct. 1998.
- [43] Q. Yu and C. Sechen, "Efficient approximation of symbolic network functions using matroid intersection algorithms", *IEEE Transactions on Computer-Aided Design*, vol. 16, pp. 1073-1081, October 1997.
- [44] O. Guerra, J.D. Rodríguez-García, E. Roca, F.V. Fernández and A. Rodríguez-Vázquez, "A simplification before and during generation methodology for symbolic large-circuit analysis," in *IEEE International Conference on Electronics, Circuits and Systems*, 1998, vol. 3, pp. 81-84.